University of Arkansas
Department of Computer Science and Computer Engineering
2009 High School Programming Contest
Problems

Wing Ning Li   Brian Lewis   Ian Taylor   Ebony Buckley

Revision Date: March 7, 2009

Please carefully read the output specification. Submitted solutions are graded by a differential program, i.e., the output produced by a submission will be compared *byte-for-byte* with the output of the judges' solutions. Thus, even though submitted solutions may produce output that is *technically* correct, it will be counted as incorrect if these specifications are not followed exactly.

For all problems, solutions receive input from standard input (*stdin*) and produce output on standard output (*stdout*).

Restricting input and output to stdin and stdout does not prevent a program from processing files. Let `Main` be the program name, `Input` be the input file name, and `Output` be the output file name. The following command at the terminal allows the program to read and write the input and output files respectively:

```
Main < Input > Output      # for C or C++, assuming Main is the executable
java Main < Input > Output # for Java, assuming Main is a class file
```

For all problems, an *integer* is defined as an optional minus sign followed by one digit from the set 1..9 followed by zero or more digits from the set 0..9. `3`, `-21`, and `309` are integers; `0123`, `- 4`, and `+38` are not.

For all problems, a *line* is defined as zero or more characters followed by a newline ('\n') character. Unless otherwise directed, a line of output should have no leading or trailing whitespace. A program producing the integer 27 on one line of output should consist of the character '2' followed by the character '7' followed by the newline ('\n') character.

1 Stock Market: Finding the Maximum Gain and Loss of a Stock

The share price of a stock can fluctuate over the course of a day. Every trading day, web sites report the intra-day high, intra-day low, opening price, and closing price of each stock.

Let a stock's closing price of some day be P and the closing price of the same or a later day be Q. The change in price between the two days is $Q - P$, of which a positive value means gain and a negative value means loss. The value of the gain or loss may be 0.

The *maximum gain* is the largest gain between two days within a period. Similarly, the *maximum loss* is the largest loss between two days within a period.

Given closing prices over a period of time, determine the maximum gain and loss.

input each line represents the closing share price of a particular stock on the provided day; lines do not occur in any particular order; fields of a line are separated by one or more space; dates follow the pattern $M/D/Y$ where M, D, and Y are all positive integers separated by a '/'.

output two lines; the first contains the maximum gain; the second line contains the maximum loss; each is formatted as in the examples below

Example:

Input

```
1/5/2009    56.52
1/20/2009   50.56
1/12/2009   51.39
1/15/2009   51.35
1/2/2009    57.18
1/14/2009   51.56
1/9/2009    51.58
1/16/2009   51.56
1/13/2009   52.12
1/8/2009    51.38
1/7/2009    55.54
1/6/2009    56.02
```

Output

```
MAXIMUM GAIN 0.74 FROM 1/2/2009 TO 1/20/2009
MAXIMUM LOSS 6.62 FROM 1/2/2009 TO 1/20/2009
```

2 Stock Market: Finding the Maximum Gain and Loss of Many Stocks

A *ticker symbol* is a group of letters that uniquely identifies publicly traded shares of a corporation. For example, the ticker symbol for Wal-Mart Stores Inc. is `WMT`. The ticker symbol for JB Hunt Transportation services Inc. is `JBHT`.

In a similar context to the previous problem, each line of input has three pieces of information: ticker symbol, date, and closing price.

Given closing prices for multiple stocks, determine the maximum gain and loss of each.

input each line represents the closing price of a particular stock on the provided day; lines do not occur in any particular order; fields of a line are separated by one or more spaces

output two lines for each stock; first line of each set contains the maximum gain; the second line of each set contains the maximum loss; each line is formatted as in the examples below; stocks are ordered by ticker symbol in lexicographical (alphabetical) order

Example:

Input

```
JBHT    1/29/2009   22.51
JBHT    1/28/2009   22.68
WMT     1/5/2009    56.52
WMT     1/20/2009   50.56
JBHT    1/22/2009   20.58
JBHT    1/21/2009   20.69
JBHT    1/20/2009   20.03
JBHT    1/16/2009   21.77
WMT     1/14/2009   51.56
WMT     1/9/2009    51.58
WMT     1/16/2009   51.56
WMT     1/13/2009   52.12
WMT     1/8/2009    51.38
WMT     1/7/2009    55.54
WMT     1/6/2009    56.02
JBHT    1/30/2009   22.27
WMT     1/12/2009   51.39
WMT     1/15/2009   51.35
WMT     1/2/2009    57.18
JBHT    1/27/2009   21.34
JBHT    1/26/2009   20.23
JBHT    1/23/2009   20.27
JBHT    1/15/2009   22.94
JBHT    1/14/2009   22.14
JBHT    1/13/2009   22.68
JBHT    1/12/2009   23.22
JBHT    1/9/2009    24.4
```

Output

```
JBHT MAXIMUM GAIN 2.65 FROM 1/9/2009 TO 1/30/2009
JBHT MAXIMUM LOSS 4.37 FROM 1/2/2009 TO 1/20/2009
WMT MAXIMUM GAIN 0.74 FROM 1/2/2009 TO 1/20/2009
WMT MAXIMUM LOSS 6.62 FROM 1/2/2009 TO 1/20/2009
```

3 Money's Future Value: Saving for Retirement

The future value of money is a very important notion in finance and it relates to everyday life in a very practical way. The parameters that affect the future value are interest rate and rate of inflation. For simplicity, this problem assumes the rate of inflation is zero. The notion of money's future value is related to the annual interest rate.

Suppose presently an account has M amount of cash and an annual interest rate of $i\%$. One year later, the account will contain $M(1 + \frac{i}{100})$ amount of cash. Following the same logic, after two years the account will have $M(1 + \frac{i}{100})^2$ (multiplying the future value after one year by $1 + \frac{i}{100}$). The above assumes that the interest is credited at the end of each year, i.e., the interest credit period is yearly. If the interest is credited at the end of each quarter the future value is calculated somewhat differently. Since each year has 4 quarters, the quarterly interest rate is derived from yearly interest as $j = \frac{i}{4}$. After one quarter, the account will have $M(1 + \frac{j}{100})$; after two quarters, $M(1 + \frac{j}{100})^2$; and after a year, $M(1 + \frac{j}{100})^4$. If the interest is credited monthly, at the end of each month, the monthly interest rate is derived from the yearly interest rate as $k = \frac{i}{12}$ and the future value of M after one year is $M(1 + \frac{k}{100})^{12}$.

One suggested approach of saving for retirement is to regularly deposit the same amount each year, quarter, month, or paycheck for a certain savings period. Given a retirement savings plan, compute the future value of money.

The interest rate is given as a yearly interest rate and is assumed to be the same throughout the savings period (it may be viewed as an approximation of the average rate). The savings period is given as one or more years. The deposit amount is given as a whole dollar amount and remains the same for each deposit. The frequency of deposit is given as either yearly, quarterly, monthly, or biweekly (assuming each year has 48 weeks, equates to 24 per year). If the frequency of deposit is yearly, the interest is credited yearly and the first deposit is made at the end of the first year. Similarly, if frequency of deposit is quarterly, the interest is credited quarterly and the first deposit is made at the end of the first quarter and so on.

Note that in the first example, the total money put in is $20 \times 12 \times 100 = 24,000$ and the money out is $98,925$. The difference is $74,925$!

input four lines; first line contains a positive decimal value representing the yearly percentage interest rate (it may be viewed as an approximation of the average rate); second line contains a positive integer representing the savings period (in years); third line contains a positive integer representing the deposit amount; fourth line contains either `yearly`, `quarterly`, `monthly`, or `biweekly` representing the frequency of deposits

output one line containing the future amount truncated to a whole dollar amount (i.e. 100.34 becomes 100 and 100.99 becomes 100); **to get an accurate result, truncation should be done at the very end!**

Example:

Input

```
12
20
100
monthly
```

Output

```
98925
```

Example:

Input

```
12
20
1200
yearly
```

Output

```
86462
```

Example:

Input

```
12
20
50
biweekly
```

Output

```
99574
```

Example:

Input

```
12
1
300
quarterly
```

Output

```
1255
```

4 Money's Present Value: Retirement Annuity

In finance and insurance, the term *annuity* is defined as a stream of fixed payments at equal intervals (i.e monthly) over a period of time. For example, getting $\$3,000.00$ each month for twenty years is an annuity.

Each fixed payment of an annuity can be viewed as the future value of money presently had. The relation of the present value of money to its future value is exactly the same as in the previous problem. For example, let M be the present value, $i\%$ be the yearly interest rate, and F be the future value of M after a year. We have $M(1+\frac{i}{100}) = F$ and $M = \frac{F}{1+\frac{i}{100}}$. To get the future value F, presently, an amount of $\frac{F}{1+\frac{i}{100}}$ is needed. Notice that *the present value of money is derived from its future value.* Similarly, to get the future value of F two years later, presently an amount of $\frac{F}{(1+\frac{i}{100})^2}$ is needed. The above assumes *the interest credit period* is yearly. If the interest credit period is monthly (credited each month at the end of the month), to get a future value of F a year later, the present value of money should be $\frac{F}{(1+\frac{k}{100})^{12}}$, where $k = \frac{i}{12}$.

Compute the present value of a given annuity, i.e., the money amount needed to generate the given retirement annuity.

To simplify the matter a bit, it is assumed the annuity payment interval is some multiple of the interest credit period and an annuity period is also some multiple of annuity payment interval. In the event that the relationship of multiplicity does not hold for the input, `INVALID` should be the output.

The first payment is made at the end of the first interval.

input five lines; line one contains a positive decimal value representing the yearly interest rate; line two contains one of the strings `yearly`, `quarterly`, or `monthly` representing the credit period; line three contains a positive integer representing the annuity period in years; line four contains a positive integer representing the payment amount; line five contains a positive integer representing the payment interval in months (e.g. 1 means every month, 2 means every two months)

output one line containing either the string `INVALID` or the present value amount; the output of the present value should be to the *next* closest dollar amount (i.e. 100.34 becomes 101 and 100.99 becomes 101)

Example:

Input

```
12
yearly
1
3000
12
```

Output

```
2679
```

Example:

Input

```
12
monthly
20
3000
1
```

Output

```
272459
```

Example:

Input

```
12
monthly
20
36000
12
```

Output

```
257797
```

Example:

Input

```
12
quarterly
20
36000
2
```

Output

```
INVALID
```

Example:

Input

```
12
monthly
20
36000
9
```

Output

```
INVALID
```

5 Prefix Notation: Evaluation

Typically, an arithmetic expression is given in infix notation, such as $10 + 70$. There is another notation called prefix, which also can be used to denote arithmetic expressions. As the name suggests, in prefix notation the operator is before the operands. The infix expression $10+70$ becomes $+ \ 10 \ 70$ (a space is used to separate the two operands 10 and 70 and the operator). A nice property of prefix is not having to worry about operator precedence, therefore eliminating the need for parentheses. Note that $a+b*c$ and $(a+b)*c$ are different in infix due to operator precedence. In prefix, $a+b*c$ becomes $+ \ a \ * \ b \ c$, and $(a+b)*c$ becomes $* \ + \ a \ b \ c$ (note that the parentheses are gone).

Our task is to write a program to evaluate an arithmetic expression in prefix notation that involves integer operands and the following binary operators $+$, $-$, $*$.

input one line containing the prefix expression; the line will not be empty; integers and operators are separated by a single space

output one line containing the integer result

Example:

Input

```
* - + 10 5 - 2 3 2
```

Output

```
32
```

6 Prefix Notation: Conversion

The previous problem asks to evaluate a prefix expression. In this problem we are asked to convert a prefix expression to an equivalent infix expression. The operands are single character lower or upper case letters, and the operators are binary operators $+$, $-$, $*$. Our program will produce a fully parenthesized infix. For example, the input prefix $+a*bc$ will result $(a+(b*c))$ as a fully parenthesized infix. Similarly, for prefix $*+abc$ we will have $((a+b)*c)$.

input one line containing the prefix expression; the line will not be empty

output one line containing the corresponding fully parenthesized infix expression

Example:

Input

```
*-+ab-CAd
```

Output

```
(((a+b)-(C-A))*d)
```

7 Coin Changing

The basic question of the coin changing problem is: using available coins, can change be made? For example, a cashier needs to provide change of 48 cents and gives the following coins: 1 quarter, 1 dime, 2 nickels, and 3 pennies. Assuming there is an unlimited supply of coins of each denomination, the cashier can make change of any amount due to having an unlimited number of pennies.

In a generalization of the coin changing problem, given a list of coin values in cents $c_1, c_2, \ldots, c_n$ and the change value K, determine whether it is possible to make change of K using the available coin types. There are an unlimited number of coins for each given coin type.

input two lines; line one contains one or more positive integers separated by a space representing the available coin types; the second line contains a positive integer representing the change value

output one line containing `YES` if it is possible to make change; `NO` otherwise

Example:

Input

```
1 5 10 25
48
```

Output

```
YES
```

Example:

Input

```
5 10 25
48
```

Output

```
NO
```

Example:

Input

```
10 5 25
50
```

Output

```
YES
```

8 Minimum Coin Changing

This problem is similar to the previous problem. If change can be made, there could be many different ways. For example, another way to provide the change of 48 cents is to use 1 quarter, 2 dimes, and 3 pennies (a total of 6 coins instead of a 7 coin solution given in the previous problem). Rather than specifying `YES` as in the previous problem, the program should output the minimum number of coins needed to make change.

input two lines; line one contains one or more positive integers separated by a space representing the coins $c_1, c_2, \ldots, c_n$; the second line contains a positive integer representing K

output one line containing a positive integer representing the minimum number of coins needed, or `NO` if making change is not possible

Example:

Input

```
1 5 10 25
48
```

Output

```
6
```

Example:

Input

```
5 10 25
48
```

Output

```
NO
```

Example:

Input

```
10 1 25
30
```

Output

```
3
```